**Project Report**

**E-commerce App with End-to-End DevOps Pipeline and Monitoring**

**Submitted by: Ashlee Agarwal**

**Pratiksha**

**Ashna**

**Yuvraj Dixit**

**Project Overview**

This project aims to design and deploy a **scalable e-commerce frontend** using React (Vite) with a **modern DevOps pipeline** for **automated, reliable deployment and monitoring**.

**Key Goals:**

Build a **dark-themed, clean UI React e-commerce frontend**.  
 Containerize using **Docker** for environment consistency.  
 Provision infrastructure using **Terraform** on AWS EC2.  
 Automate deployment with **Jenkins CI/CD pipeline**.  
 Monitor system and application health using **Prometheus and Grafana**.

**Technology Stack**

| **Layer** | **Technologies** |
| --- | --- |
| Frontend | React (Vite), Normal CSS |
| Containerization | Docker |
| Infrastructure as Code | Terraform |
| CI/CD | Jenkins |
| Cloud | AWS EC2 |
| Monitoring | Prometheus, Grafana |
| SCM | Git, GitHub |

**Implementation Details**

**A. Frontend Development**

* Used **Vite** for fast React build tooling.
* Designed **dark-themed e-commerce UI** with:
  + Home page showcasing products.
  + Product details page.
  + Navigation and footer.
* Styled using **clean CSS, no frameworks**, ensuring responsiveness and aesthetic readability.

**B. Containerization with Docker**

* Created a **multi-stage Dockerfile**:
  + Stage 1: Build Vite React app.
  + Stage 2: Use Nginx to serve built files.
* Configured nginx.conf for **SPA routing** and performance.

**C. Infrastructure Provisioning using Terraform**

* Wrote Terraform scripts to:
  + Provision AWS EC2 instance with Docker installed.
  + Configure security groups for HTTP (80) and SSH (22).
  + Use SSH key for secure access.

**D. CI/CD Pipeline using Jenkins**

* Configured **Jenkins pipeline using Jenkinsfile**:
  + **Clone repo on push.**
  + Build Docker image and push to Docker Hub.
  + SSH into EC2, pull image, stop old container, run new container automatically.
* Uses:
  + Docker Pipeline Plugin.
  + SSH Agent Plugin for EC2 deployment.
  + Docker Hub credentials securely managed.

**E. Monitoring with Prometheus and Grafana**

* Used **Prometheus** to collect metrics.
* Used **Grafana** for visualization dashboards.
* Configured with docker-compose for easy monitoring stack deployment on EC2.
* Monitored:
  + Container uptime.
  + CPU and RAM usage.
  + Health checks for app availability.

**Workflow**

**Developer pushes code** →  
 **GitHub triggers Jenkins pipeline** →  
 **Jenkins builds Docker image and pushes to Docker Hub** →  
 **Jenkins SSH deploys container on EC2 using Docker** →  
 **Prometheus + Grafana monitor deployment health.**

* Home page of the e-commerce app.
* Product page.
* Jenkins pipeline stages.
* Grafana dashboard.
* Prometheus metrics page.
* Terraform apply output with EC2 instance IP.

**Challenges Faced**

* Compatibility issues with **Node.js and Vite** resolved by using Node 20 LTS.
* Managed **merge conflicts** during GitHub push.
* Configuring Nginx for correct SPA routing.
* Setting up secure SSH pipelines for Jenkins deployment.
* Configuring Prometheus scrape targets correctly for EC2 monitoring.

**Key Learnings**

End-to-end **CI/CD automation** with Docker, Jenkins, and Terraform.  
 **Infrastructure provisioning** using Terraform, understanding state management and security group configuration.  
 Deployment of **React apps using Docker & Nginx** for scalable delivery.  
 **Container and app monitoring** using Prometheus and Grafana for production-grade observability.  
 Effective **version control workflows** using Git and GitHub.

**Future Scope**

Add backend APIs using **Node/Express + MongoDB** for full-stack capability.  
 Implement **JWT authentication** for user login.  
 Enable **payment gateway integration** for real e-commerce functionality.  
 Add **HTTPS SSL certificates** using Let’s Encrypt for secure delivery.  
 Transition to **Kubernetes (EKS) for auto-scaling** under load.  
 Automate using **GitHub Actions for CI/CD** for serverless pipeline management.

**Conclusion**

This project demonstrates a **production-grade DevOps workflow for frontend applications** using modern tooling, ensuring **reliable, automated, and observable deployments**.

It builds a solid foundation to extend into **full-stack applications and scalable production pipelines** while adhering to best practices in modern software engineering.